<tiger2/> as a standardised serialisation for ISO 24615 – SynAF

Sonja Bosch, University of South Africa (UNISA) in Pretoria – Key-Sun Choi, KAIST – Éric de la Clergerie, Inria – Alex Chengyu Fang, City University of Hong Kong – Gertrud Faaß, University of Hildersheim – Kiyong Lee, Korea University – Antonio Pareja-Lora, Universidad Complutense de Madrid – Laurent Romary, Inria & Humboldt University – Andreas Witt, Institut für Deutsche Sprache – Amir Zeldes, Humboldt Universität zu Berlin – Florian Zipser, Inria & Humboldt Universität zu Berlin

Abstract

This paper presents the application of the <tiger2/> format to various linguistic scenarios with the aim of making it the standard serialisation for the ISO 24615 [1] (SynAF) standard. After outlining the main characteristics of both the SynAF metamodel and the <tiger2/> format, as extended from the initial Tiger XML format [2], we show through a range of different language families how <tiger2/> covers a variety of constituency and dependency based analyses.

1 From SynAF to <tiger2/>

In 2010, the International Organization for Standardization (ISO) published ISO 24615 (SynAF), which provides a reference model for the representation of syntactic annotations. This document elaborates on early proposals for the representation of syntactic information [3] and [4] to provide a comprehensive framework based on the following principles:

- Dealing with the representation of both syntactically annotated corpora and data resulting from an automatic syntactic analysis
- Equally covering dependency and constituency based representations
- Being flexible enough to deal with a variety of languages by relying on an open metamodel which can be parameterized by means of specific data categories

Still, the SynAF standard did not contain any specific XML serialisation. This prevented it from being sufficient for implementers who would want to ensure that their data be interoperable with other initiatives, at least for the same language. This is why ISO subcommittee TC 37/SC 4 initiated a new activity, in order to provide an additional part to SynAF on the basis of the Tiger XML format [2]. This format has been slightly adapted[1] in the

[1] All background information on <tiger2/> can be found at http://korpling.german.hu-berlin.de/tiger2/, comprising schemas, documentation and encoded examples.

meantime under the code name <tiger2/>, to comply with recent XML technology developments[2], to integrate some necessary features and to fully encompass dependency-based representations.
Before proceeding any further with this initiative, the SC4 experts who attended the ISO TC37 meeting in Madrid in June 2012 decided to put <tiger2/> to the test, by applying it to a variety of language samples. This should provide a concrete basis for the future committee work, which should integrate the feedback from the community interested in syntactic annotation and in particular colleagues involved in the development of treebanks. In this context, this paper is intended to provide a first overview of these experiments, with the aim of raising some interest and thereby getting some feedback from a variety of interested experts.
This paper will first of all describe the major characteristics of both the ISO SynAF standard and the <tiger2/> formats and then present a series of short sections demonstrating a) the possible application of <tiger2/> in specific linguistic contexts; and b) how SynAF can be used as an interchange format for existing tools. As mentioned, this paper discusses work in progress. Consequently, we expect it to lead other colleagues to start experimenting with <tiger2/> and, thus, provide some concrete feedback to the forthcoming standardisation work.

2 The SynAF metamodel

ISO 24615 (SynAF) provides a high-level metamodel for representing the syntactic annotation of linguistic data, with the objective of supporting interoperability across language resources or language processing components. SynAF aims at covering the two main functions of syntactic annotation in language processing, namely: a) to represent linguistic constituency, as in Noun Phrases (NP) etc., describing a structured sequence of morpho-syntactically annotated items (including, depending on the theory used, empty elements or traces generated by movements at the constituency level), as well as constituents built from non-contiguous elements; and b) to represent dependency relations, such as head-modifier relations, and also including relations between categories of the same kind (such as the head-head relations between nouns in appositions, or nominal coordinations in some formalisms).
As a consequence, syntactic annotations must comply with a multi-layered annotation strategy, which interrelates syntactic annotation for both

[2] Typically: generic XML attributes, schema technology, XML design good practices. Although it is not the aim of this paper to go in to the corresponding technical details, we encourage interested readers to refer to the <Tiger2/> documentation

constituency and dependency, possibly simultaneously, as stated in the SynAF metamodel. The metamodel is based on the following components:

- The *T_node* component represents the terminal nodes of a syntactic tree, consisting of morpho-syntactically annotated word forms, as well as empty elements when appropriate. T_nodes are annotated with syntactic categories valid for the word level.
- The *NT_node* component represents the non-terminal nodes of a syntactic tree. The NT_nodes are annotated with syntactic categories that are valid at the phrasal, clausal and/or sentential levels.
- The *Edge* component represents a relation between syntactic nodes (both terminal and non-terminal nodes). For example, the dependency relation is binary, consisting of a pair of source and target nodes, with one or more annotations.

From this metamodel, a specific syntactic annotation model can be obtained by combining the above-mentioned components with data categories characterising or refining their semantics.

It should be noticed that the terminal node level in SynAF is strongly equivalent to the word form level in MAF (ISO 24611 [5]), for which we offer concrete interfaces below. It is thus left to the implementer to either separate or merge these two components depending on whether it is relevant or not, for instance, to clearly differentiate the data categories attached to word forms and terminals within a multi-layered annotated corpus.

3 <tiger2/> as a SynAF compliant Tiger

The main characteristics of the <tiger2/> datamodel can be summarized as follows:

- Terminal nodes are implemented as <t> elements, either referring to a textual segment or pointing to a word form in a morphosyntactically annotated corpus. Non-terminal nodes are implemented as <nt> elements and are used to represent hierarchical structures like syntactic trees
- The Edge component is implemented by means of an <edge> element, which may relate either <nt> or <t> elements to each other.
- A generic @type attribute inside terminal, non-terminal and edge elements can further qualify the meaning of these elements. For instance a non-terminal node can be qualified as a verbal complex, a terminal can be specified as a compound node etc.
- The elements terminal, non-terminal and edge as well are used as placeholders for further linguistic annotations. Such annotations can freely be added as generic attribute-value-pairs with no restrictions to

a specific linguistic school or theory (e.g. attributes like @cat, @phrase, … or any user defined attributes).

- In addition to corpus metadata, an annotation block at the beginning of the document allows to declare the particular descriptive features used in the annotation project.
- An important change from the initial Tiger format [2] is that with <tiger2/> it is possible to directly implement edges between terminal modes, thus empowering the datamodel with a fully-fledged dependency-based representation. As demonstrated in section 4.2 for Semitic languages, such edges may appear as children of <t> elements. They may also be used for further sentential and even inter-sentential relations such as binding and anaphora (see also subsections 4.1.4 and 4.3.5).

All these features have been implemented on the basis of a complete specification written in the TEI ODD language from which one can generate both various schemas (DTD, RelaxNG, W3C) and the actual documentation. Since not all the features integrated in <Tiger2/> may be needed at the same time for a specific implementation, it is anticipated that project specific customizations, based on the ODD developed made so far, will allow the community to converge on specific flavours of <Tiger2/>.

4 <tiger2/> at work – linguistic cases

At this stage, we do not want to discuss the actual coverage and expressive power of the <tiger2/> proposal further without validating it across a variety of possible languages. To this end, the authors of this paper have volunteered to test the <tiger2/> proposal on the languages where they had the appropriate expertise. This is reflected in this section, where a quick overview of the main characteristics and challenges is given for a sample of European, African and Asian languages, followed by the explicit representation of an example in <tiger2/>. Even if the actual validation of the model will only occur when wide-coverage syntactic analyses or treebanks will be transformed into this format[3], it will allow us to estimate in which direction the work on the future second part of SynAF should go.

4.1 Spanish and Romance Languages

The set of syntactic phenomena that are present in Romance languages do not differ much from those present in English. Hence, their representation does not require adding many other particular devices to the ones included in

[3] and accordingly a comprehensive set of data categories for such languages will be made available in ISOCat

SynAF and/or in <tiger2/>. However, they share some common particularities that will be discussed next, namely enclitic objects and contractions, elliptic subjects and redundancy, which can be described using a combination of MAF and SynAF [6], [7] and [8]. Each of them is presented in a dedicated subsection below.

4.1.1 Spanish enclitic objects

First, most Romance languages allow clitic constructions, much like Semitic languages (see Section 4.2). For example, the Spanish sentence 'Díselo al médico' (= 'Tell [to him] it to the doctor) includes a couple of enclitic pronouns ('se' and 'lo') attached to the word form 'Dí', which is the main verb of the sentence. This phenomenon can be found also in Italian and in Catalan, amongst others. The morphosyntactic segmentation of this sentence is shown in Example 1, using the encoding scheme of MAF, which also shows the particular morphosyntactic annotation of 'Díselo'.

```
<maf>
  <token xml:id="t1">Dí</token>      <!-- Dí = Tell   -->
  <token xml:id="t2">se</token>      <!-- se = to him -->
  <token xml:id="t3">lo</token>      <!-- lo = it   -->
  <token xml:id="t4">a</token>       <!-- a = to    -->
  <token xml:id="t5">l</token>       <!-- el = the  -->
  <token xml:id="t6">médico</token>  <!-- médico = doctor -->
  <token xml:id="t7">.</token>

  <!-- wordForm for "Díselo", which is the aggregation of -->
  <!-- the verbal form "Dí" and the enclitic pronouns -->
  <!-- "se" and "lo".                                 -->
  <wordForm xml:id="wordForm1" tokens="t1 t2 t3"/>
    <!-- wordForms for "Dí", "se" and "lo" -->
    <wordForm xml:id="wordForm2" lemma="decir" tokens="t1">
      <fs>
        <f name="pos">
          <symbol value="VMI2S"/>
          <!-- Verb-Main,Imperative,2nd person,Singular -->
        </f>
      </fs>
    </wordForm>
    <wordForm xml:id="wordForm3" lemma="se" tokens="t2">
      <fs>
        <f name="pos">
          <symbol value="PP3S"/>
          <!-- Personal Pronoun, 3rd person, Singular -->
         </f>
      </fs>
    </wordForm>
    <wordForm xml:id="wordForm4" lemma="lo" tokens="t3">
      <fs>
        <f name="pos">
          <symbol value="PP3S"/>
          <!-- Personal Pronoun, 3rd person, Singular -->
        </f>
      </fs>
    </wordForm>
  </wordForm>
  <!-...->
```

```
</maf>
```

Example 1: Morphosyntactic segmentation of the Spanish sentence 'Díselo al médico' and morpho-syntactic annotation of the clitic construction 'Díselo'

4.1.2 Contractions

Second, most Romance languages include some contractions, that is, some word forms consisting of the fusion of a preposition and an article. This is not particular to Romance languages, since English and German (for example) include this type of words too. Example 2 shows how the Spanish contraction 'al' (= 'to the', which corresponds also to 'au' in French, for instance) has been annotated conforming to MAF.

```
<maf>
  …
  <!-- wordForm for "al", which is the contraction of  -->
  <!—the preposition "a" and the definite article "el" -->
  <wordForm xml:id="wordForm5" tokens="t4 t5"/>

    <!-- wordForms for "a" and "el" -->
    <wordForm xml:id="wordForm6" lemma="a" tokens="t4"/>
      <fs>
        <f name="pos">
          <symbol value="AP"/>      <!-- Adposition (Preposition) -->
        </f>
      </fs>
    <wordForm xml:id="wordForm7" lemma="el" tokens="t5"/>
      <fs>
        <f name="pos">
          <symbol value="ATDMS"/>
          <!-- Article-Definite, Masculine, Singular -->
      </f>
      </fs>
  </wordForm>
  …
</maf>
```

Example 2: Morphosyntactic annotation of the Spanish contraction 'al'

4.1.3 Elliptic subjects

Third, Romance languages are characterized by the agreement in person and in number between the subject and the main verb of the sentence, and some of them (like Spanish) also require the subject and the main verb to agree in gender in certain cases. Besides, the inflection of Spanish verbs usually identifies unambiguously the person and the number of the subject. Due to this, Spanish subjects are elliptic in most cases. In fact, if the subject is made explicit in such cases, it can be considered redundant and, hence, pragmatically emphasized. Therefore, a mechanism for making elliptic subjects explicit might be required in order to annotate the full structure and/or set of dependencies of Spanish sentences. This mechanism would be similar to the one shown in Example 1 and 2 for Buntu (see 4.3.5), which

shows how some elements that are not present on the surface structure can be made explicit by means of the standard.

4.1.4 Grammar-required redundancy

Fourth, the enclitic constructions mentioned in Section 4.1.1 usually entail the inclusion of a redundant syntactic element, like the pronoun ‘se’ in the Spanish sentence ‘Díselo al médico’. This a personal pronoun, which coreferences the nominal phrase ‘al médico’ and is, thus, redundant. Nevertheless, removing this pronoun would produce a wrong (that is, ungrammatical) sentence. So a coreference mechanism has to be allowed in order to tag this type of redundancies. This can be done also following the same mechanism used in Example 1 and 2, as shown in Example 3. The annotation associated to redundancy has been highlighted for convenience. The example only includes the section for terminals for the sake of space.

```
<corpus ...>
 <head>
  <!-- ... -->
  <meta>
   <name>spanish</name>
  </meta>
  <annotation>
   <!-- ... -->
   <feature domain="edge" name="label" type="coref">
    <value name="Anaph">Anaphoric</value>
   </feature>
  </annotation>
 </head>
 <body>
  <s xml:id="s1">
   <graph root="s1_ROOT" discontinuous="false">
    <terminals>
     <t xml:id="s1_t1"
       tiger2:corresp="spanish.example.maf.xml#wordForm2"/>
     <!-- Decir (Di-) = Tell -->
     <t xml:id="s1_t2"
       tiger2:corresp="spanish.example.maf.xml#wordForm3">
      <edge tiger2:type="coref" label="Anaph"
          tiger2:target="#s1_nt4"/>
      <!-- se (-se-) = to him -->
     </t>
     <t xml:id="s1_t3" tiger2:corresp="spanish.maf.xml#wordForm4"/>
     <!-- lo (-lo) = it -->
     <t xml:id="s1_t4" tiger2:corresp="spanish.maf.xml#wordForm6"/>
     <!-- a (a-) = to -->
     <t xml:id="s1_t5" tiger2:corresp="spanish..maf.xml#wordForm7"/>
     <!-- el (-l) = the -->
     <t xml:id="s1_t6" tiger2:corresp="spanish.maf.xml#wordForm8"/>
     <!-- médico = doctor -->
     <t xml:id="s1_t7" tiger2:corresp="spanish.maf.xml#wordForm9"/>
     <!-- . -->
    </terminals>
   <nonterminals>
    <!-- ... -->
   </nonterminals>
  </graph>
```

```
  </s>
 </body>
</corpus>
```

Example 3: SynAF-<TIGER2/>-conformant syntactic annotation of the Spanish sentence 'Díselo al medico.'

4.2 Semitic Languages

Semitic languages bring with them a particular set of challenges for modelling linguistic data. The following two subsections demonstrate how MAF and SynAF/<tiger/2> can work together to represent two problematic constructions: dependencies for object clitics and constituents for construct state compounds.

4.2.1 Arabic enclitic objects

Much like Romance languages, both Arabic and Hebrew have the ability to express pronominal objects as enclitic pronouns that are written together with the verb, forming one word form on the orthographic level [9], e.g. in Standard Arabic:

رأيت الملك
ra’aitu l-malik ‘I saw the king’
see.PF.1.SG DEF-king

رأيته
ra’aitu-hu ‘I saw him’
see.PF.1.SG-3.SG.M

Modelling a syntax tree of the dependencies between verb and object can be difficult and usually leads to make the design decision of tokenizing enclitic object pronouns as separate word forms. Using MAF word forms it is possible to keep such orthographic strings together while referring directly to constituent elements in the SynAF representation. The following representation gives a possible solution for a unified representation of the examples above, similar to the Spanish case.

```
<maf>
  <token xml:id="s1tk1">رأيت</token>        <!-- ra’aitu = I saw-->
  <token xml:id="s1tk2">ال</token>          <!-- l = the -->
  <token xml:id="s1tk3">ملك</token>         <!-- malik = king -->
  <token xml:id="s1tk4">.</token>           <!-- . -->
  <token xml:id="s2tk1">رأيت</token>        <!-- ra’aitu = I saw -->
  <token xml:id="s2tk2">ه</token>           <!-- hu = him -->
  <token xml:id="s2tk3">.</token>           <!-- . -->

  <wordForm xml:id="s1wf1" tokens="s1tk1">          <!-- ra’aitu -->
  <wordForm xml:id="s1wf2" tokens="s1tk2 s1tk3">    <!-- l-malik -->
  <wordForm xml:id="s1wf3" tokens="s1tk4">          <!-- . -->
  <wordForm xml:id="s2wf1" tokens="s2tk1 s2tk2">    <!--ra’aitu-hu -->
  <wordForm xml:id="s2wf2" tokens="s2tk3">          <!-- . -->
</maf>
```

Example 4: MAF representation of the Arabic clitic examples above.

```
<s xml:id="s1">
 <graph>
  <terminals>
   <!-- ra'aitu = I saw -->
   <t xml:id="s1_t1" tiger2:corresp="arabic.maf.xml#s1wf1">
    <edge xml:id="s1_e1" tiger2:target="#s1_t3" tiger2:type="dep"
     label="obj"/>
   </t>
   <!-- l = the -->
   <t xml:id="s1_t2" tiger2:corresp="arabic.maf.xml#s1tk2" />
   <!-- malik = king -->
   <t xml:id="s1_t3" tiger2:corresp="arabic.maf.xml#s1tk3">
    <edge xml:id="s1_e2" tiger2:target="#s1_t2" tiger2:type="dep"
     label="det"/>
   </t>
  </terminals>
  <nonterminals/>
 </graph>
</s>
<s xml:id="s2">
 <graph>
  <terminals>
   <t xml:id="s2_t1" tiger2:corresp="arabic.maf.xml#s2tk1" >
    <edge xml:id ="s1_e1" tiger2:target="#s2_t2" tiger2:type="dep"
     label="obj"/>
   </t>
   <t xml:id="s2_t2" tiger2:corresp="arabic.maf.xml#s2tk2" />
  </terminals>
  <nonterminals/>
 </graph>
</s>
```

Example 5: <tiger2/> representation of dependency trees for the Arabic examples above.

By pointing at both word forms and tokens, the SynAF representation can contain uniform syntax trees for both enclitic and separate objects.

4.2.2 Hebrew construct states

Both Hebrew and Arabic use a special left-headed construction similar to compounding [10] to form complex nouns. This construction has a number of special features. Most notably for the present discussion, it does not constitute an orthographic word form, but it allows only one article for the entire complex noun, placed between the head and the modifier (or before the last modifier in recursive complex nouns). The article, like all articles in both languages, is written together with the final modifier and does constitute one word form with it, but marks the entire construction as definite. The following examples from Hebrew illustrate the positioning of the relevant elements in definite and indefinite environments:

בית חולים
beit xolim 'hospital'
house sick.PL

בית החולים
beit ha-xolim 'the hospital'
house the-sick.PL

As a result of this configuration, a syntactic analysis may choose to regard the interposed article as attached to the entire complex noun around it (‘hospital’), rather than the modifier to which it is attached (‘sick’). A MAF/SynAF representation reflecting this analysis is given below. Note that the MAF representation contains a discontinuous word form to represent the word for ‘hospital’ and its lemma, as well as a word form uniting ‘sick’ with the orthographically attached article, but without a lemma (since ‘the sick’ has no lexical status as a lemma).

```
<maf>
  <token xml:id="s1tk1">בית</token>         <!-- beit = house -->
  <token xml:id="s1tk2">ה</token>           <!-- ha = the -->
  <token xml:id="s1tk3">חולים</token>       <!-- xolim = sick -->

  <!-- beit ...xolim -->
  <wordForm xml:id="s1wf1" lemma="בית חולים" tokens="s1tk1 s1tk3">
  <!-- ha-xolim -->
  <wordForm xml:id="s1wf2" tokens="s1tk2 s1tk3">
</maf>
```

Example 6: MAF representation of the Hebrew construct state examples above.

```
<s xml:id="s1">
 <graph>
  <terminals>
   <t xml:id="s1_t1" tiger2:corresp="hebrew.maf.xml#s1wf1"/>
   <!-- beit...xolim = hospital -->
   <t xml:id="s1_t2" tiger2:corresp="hebrew.maf.xml#s1tk2"/>
   <!-- ha = the -->
  </terminals>
  <nonterminals>
   <nt xml:id="s1_nt1" cat="NP">
    <edge tiger2:target="#s1_t2" tiger2:type="prim" label="DT"/>
    <!-- ha = the -->
    <edge tiger2:target="#s1_t1" tiger2:type="prim" label="HD"/>
    <!-- beit...xolim = hospital -->
   </nt>
  </nonterminals>
 </graph>
</s>
```

Example 7: <tiger2/> representation of a constituent tree for the definite construct state.

This way, the definite NP behaves in a usual manner by joining a noun with an article, and the complexity of the noun is hidden in the MAF representation. Another option is to model the entire graph within the SynAF level, as in the following representation.

```
<s xml:id="s1">
 <graph>
  <terminals>
   <t xml:id="s1_t1" tiger2:word="בית"/>     <!-- beit = house -->
   <t xml:id="s1_t2" tiger2:word="ה"/>       <!-- ha = the -->
   <t xml:id="s1_t3" tiger2:word="חולים"/>   <!-- xolim = sick -->
  </terminals>
  <nonterminals>
```

```
   <nt xml:id="s1_nt1" tiger2:type="construct" lemma="בית חולים">
    <edge tiger2:target="#s1_t1" tiger2:type="const" label="HD"/>
    <!-- beit = house -->
    <edge tiger2:target="#s1_t3" tiger2:type="const" label="MO"/>
    <!-- xolim = sick -->
   </nt>
   <nt xml:id="s1_nt2" tiger2:type="phrase" cat="NP">
    <edge tiger2:target="#s1_t2" tiger2:type="prim" label="DT"/>
    <!-- ha = the -->
    <edge tiger2:target="#s1_nt1" tiger2:type="prim" label="HD"/>
    <!-- beit...xolim = hospital -->
   </nt>
  </nonterminals>
 </graph>
</s>
```

Example 8: <tiger2/> representation of the definite construct state without a MAF layer.

In this representation there are two types of non-terminals and edges: a phrase node with primary dominance edges ('prim'), and a construct node with 'const' edges to its constituents. These must be declared in the annotation block and may carry different annotations, e.g. the lemma for construct states.

4.3 Zulu

4.3.1 Zulu agreement and writing system, encoding in MAF

The Bantu language family, of which Zulu is a member, is characterized by an agreement system based on noun classification [11]. The basic word order in Zulu is SVO, although this availability of grammatical agreement allows a fairly unrestricted word order. In Zulu, a verb hence usually contains a prefix that cross-references it to the subject noun. Zulu is a language written conjunctively, i.e. linguistic units are merged to morpho-phonetic units on the surface.

4.3.2 Cross-referencing noun phrases in verb phrases

In the example sentence shown in Table 1, the *wa-* prefix in the verb *wabuya* agrees with the subject noun *umfana*. Both are of noun class 1, which usually indicates that the noun is referring to a singular human. In Table 1, the first line shows the surface sentence, the second its underlying morphs, and the third line demonstrates its part-of-speech (or rather morph) categories.

Umfana		wabuya			kodwa	akayisizanga			
Um-	fana	wa-	-buy-	-a	kodwa	aka-	-yi-	-siz-	-anga
class 1 nom. marker	noun stem	subj-3rdcl1 prefix past tense	verb root	verb ending	CONJ	neg.subj-3rd-cl1 prefix	obj-3rd-cl9 prefix	verb root	verb negation marker past tense
	boy		return		but	(he)	her	assist	not

Table 1: A brief description of the Zulu sentence 'Umfana wabuya kodwa akayisizanga' (The boy returned but he did not assist her)

While subject-verb agreement is obligatory, the verb-object phrase only shows grammatical agreement in certain cases, for instance, when the referent of the object is known information in the discourse and therefore omitted, as illustrated in the example sentence. The object-affix *–yi-* in *akayisizanga* is an example of pronominal incorporation in the verb, since it refers to the class 9 noun *intombazane* (girl), a discourse topic that has already been established. Note that, although it is a dependent morpheme, this object affix carries the grammatical function of an object (see also [12] on *Kichaga*, another Bantu-language).

4.3.3 Inflection (negation and tense indication)

One way to negate a sentence in Zulu is to modify the verb. In this case, the sentence is negated by using two verbal affixes, i.e. the prefix *aka-* and the suffix *-anga* which also indicates past tense (*-i* would have marked present tense). In the first verb, *wa-buya*, past tense is indicated by the class prefix *wa-*; here, present tense would have been indicated by the prefix *u-*.

4.3.4 Compound sentences

The sentence in Table 1 is an example of a compound sentence that links clauses of equal importance. In this case, the co-ordination is achieved by means of the co-ordinating conjunction *kodwa* (but). Our suggested encoding in MAF (the token definitions are found in Example 9) reflects the morpho-syntactic facts: some of the units consisting of several morphs are formed on word level (see Example 10), while others form the predicate, i.e. a verbal phrase containing an agreement marker, an object, a verb stem and an inflectional element, cf. (Example 11).

```
<token join="right" xml:id="t1" form="Um"/>
<!-- um: noun class prefix class 1-->
<token xml:id="t2" form="fana"/>
<!-- fana: noun stem common (boy)-->
<token join="right" xml:id="t3" form="wa"/>
<!-- wa: subject prefix class 1-->
<token xml:id="t4" form="buy"/>
<!-- buy: verb root (return) -->
<token join="left" xml:id="t5" form="a"/>
<!-- -a: verb ending/extension -->
<token xml:id="t6" form="kodwa"/>
<!-- kodwa: conjunction (but)
<token join="right" xml:id="t7" form="aka"/>
<token join="right" xml:id="t8" form="yi"/>
<token xml:id="t9" form="siz"/>
<token join="left" xml:id="t10" form="anga"/>
<token xml:id="t11" form="."/>
```

Example 9: Excerpt of the MAF-encoding: token definitions

```
<wordForm xml:id="wordForm10" tokens="t1 t2" lemma="umfana">
 <wordForm tokens="t1">
  <fs>
   <f name="pos">
    <symbol value="morph.CP class.01 pers.01 num.sg"/>
   </f>
  </fs>
 </wordForm>
 <wordForm>
  <fs>
   <f name="pos"> <symbol value="morph.NSC"/> </f>
  </fs>
 </wordForm>
</wordForm>
```

Example 10: Excerpt of the MAF-encoding: morphs forming a noun

```
<!-- the wordform consists of four wordForms, the first two and -->
<!-- the last one are grammatical morphemes. -->
<!-- These will be put together in Synaf because the result is -->
<!-- a syntactic category rather than a word -->
<wordForm xml:id="wordForm40" tokens="#t7">
 <fs>
  <f name="pos">
   <symbol value="morph.CS class.01 pers.03 num.sg pol.neg"/>
  </f>
 </fs>
</wordForm>
<wordForm xml:id="wordForm41" tokens="#t8">
 <fs>
  <f name="pos">
   <symbol value="morph.CO class.09 pers.03 num.sg"/>
  </f>
 </fs>
</wordForm>
<wordForm xml:id="wordForm42" tokens="#t9" lemma="siza">
 <fs>
  <f name="pos"><symbol value="morph.VR_tr"/></f>
 </fs>
</wordForm>
<wordForm xml:id="wordForm43" tokens="#t10">
 <fs>
  <f name="pos"><symbol value="morph.neg" tense="past"/></f>
 </fs>
</wordForm>
```

Example 11: Excerpt of the MAF-encoding: units of the predicate

4.3.5 Encoding in SynAF

For space reasons, we focus on two of the phenomena described above: defining an anaphoric target in the second clause (see Example 12), and defining the predicate as a syntactic unit (a VP, see Example 13) formed by surface linguistic tokens, some of which are dependent morphs and others are free morphemes, i.e. lexical units.

```
<terminals>
 <t xml:id="s1_t1"
    tiger2:corresp="example.standoff.maf.xml#wordForm10"/>
 <!-- umfana -->
 <!-- ... -->

 <t xml:id="s1_t6" tiger2:type="PRO" function="SC">
  <!-- (umfana) -->
  <edge tiger2:type="coref" label="anaphoric" tiger2:target="#s1_t1"/>
 </t>
</terminals>
```

Example 12: Excerpts of our SynAF-encoding demonstrating the capability of the standard to represent anaphoric elements, non-existent on the surface.

```
<!--...-->
<!-- aka-yi-siz-anga -->
<t xml:id="s1_t7"
   tiger2:corresp="example.standoff.maf.xml#wordForm40"/>
<t xml:id="s1_t8"
   tiger2:corresp="example.standoff.maf.xml#wordForm41"/>
<t xml:id="s1_t9"
   tiger2:corresp="example.standoff.maf.xml#wordForm42"/>
<t xml:id="s1_t10"
   tiger2:corresp="example.standoff.maf.xml#wordForm43"/>
<!--...-->
</terminals>
<!--...-->
<non-terminals>
 <nt xml:id="s1_nt400" cat="VP01">   <!-- aka-yi-siz-anga -->
  <edge tiger2:type="prim" tiger2:target="#s1_t7"/>
  <edge tiger2:type="prim" tiger2:target="#s1_t8"/>
  <edge tiger2:type="prim" tiger2:target="#s1_t9"/>
  <edge tiger2:type="prim" tiger2:target="#s1_t10"/>
  </nt>
</non-terminals>
```

Example 13: Description of terminal units that together appear as a syntactic element (the predicate)

4.4 Chinese

The following encodings illustrate the treatment of an example sentence in Chinese. The sentence reads "她们正在学习古代汉语。" ("*They are currently studying classical Chinese.*"). It contains a total of 11 Chinese characters including the sentence-end full stop, which are segmented into 6 lexical units as terminals. Syntactically speaking, the string, like most other sentences in Chinese, exhibits an SVO construction, with "她们" (*They*) as the subject NP, "学习" (*studying*) as the verb and "古代汉语" (*classical*

Chinese) as the direct object NP. “正在” (*currently*) is analyzed as an adverb, forming part of the VP.

There is no explicit tense marking in the sentence. The progressive aspect is lexically expressed through the adverb “正在” (*currently*), reaffirming the fact that there is no morphological change for the Chinese verb. As a matter of fact, except for a few dependent morphemes indicating plurality, Chinese nouns do not have inflectional changes. In this sense, the attribute “lemma” in <terminals> below is not really necessary. In addition, the plural personal pronoun “她们” (*they*) is feminine, illustrating one of the very few personal pronouns carrying gender information. As a general rule, there is no gender marking for Chinese nouns. It is thus questionable whether gender marking should be a default attribute. In the current example, such marking is omitted for the sake of simplicity and clarity.

```
<s xml:id="s1">
 <graph root="s1_ROOT" discontinuous="true">
  <terminals>
   <t xml:id="s1_t1" tiger2:word="她们" lemma="她们" pos="PP"/>
   <t xml:id="s1_t2" tiger2:word="正在" lemma="正在" pos="RB"/>
   <t xml:id="s1_t3" tiger2:word="学习" lemma="学习" pos="VB"/>
   <t xml:id="s1_t4" tiger2:word="古代" lemma="古代" pos="JJ"/>
   <t xml:id="s1_t5" tiger2:word="汉语" lemma="汉语" pos="NN"/>
   <t xml:id="s1_t6" tiger2:word="。" lemma="。" pos="."/>
  </terminals>
  <nonterminals>
   <nt xml:id="s1_nt1" cat="NP">
    <edge tiger2:target="#s1_t1" tiger2:type="prim" label="HD"/>
    <!--她们-->
   </nt>
   <nt xml:id="s1_nt2" cat="VP">
    <edge tiger2:target="#s1_t2" tiger2:type="prim"/>
    <!-- 正在 -->
    <edge tiger2:target="#s1_t3" tiger2:type="prim" label="HD"/>
    <!-- 学习 -->
    <!-- labels can be ommited as in the next edge -->
    <edge tiger2:target="#s1_nt3" tiger2:type="prim" label="DO"/>
    <!-- NP -->
   </nt>
   <nt xml:id="s1_nt3" cat="NP">
    <edge tiger2:target="#s1_t4" tiger2:type="prim" label="AT"/>
    <!-- 古代 -->
    <edge tiger2:target="#s1_t5" tiger2:type="prim" label="HD"/>
    <!-- 汉语 -->
   </nt>
   <nt xml:id="s1_nt4" cat="S">
    <edge tiger2:target="#s1_t1" tiger2:type="prim" label="SB"/>
    <!--她们 -->
    <edge tiger2:target="#s1_nt2" tiger2:type="prim"/>
    <!-- VP -->
   </nt>
   <nt xml:id="s1_ROOT" cat="ROOT">
    <edge tiger2:target="#s1_nt2" tiger2:type="prim"/>
    <!-- VP -->
    <edge tiger2:target="#s1_t6" tiger2:type="prim"/>
    <!-- 。 -->
   </nt>
```

```
  </nonterminals>
 </graph>
</s>
```

Example 14: example sentence in Chinese

4.5 Korean

Korean is known to be an *agglutinative head-final* language, much like Japanese (see [13] and [14] for details).

4.5.1 Agglutination

Basic sentential segments, separated by whitespace, are called *eojeol* in Korean. Each eojeol is formed through a series of so-called *agglutination* processes. Each agglutination process concatenates a stem with a suffix, either nominal or predicate, and then this process may repeat, as in [[[타]verbStem-[쓰]pas]stem-[던]rx]eojeol.

미아가 탔던 차
mia.ga that.deon cha
Mia+NOM use+PAS+RX car
'car (which) Mia used to ride'

where NOM is a nominative case marker, PAS a past tense marker, and RX a relativizing (retrospective) suffix, called *adnominalizer*.

This string consists of three *eojeol*: (eo1) 미아가, (eo2) 탔던, and (eo3) 차. Some eojeol, such as (eo2) and (eo3), are word forms by themselves, whereas the first eojeol (eo1) is segmented into two word forms, 미아 'Mia' and –가, a nominative case marker. Each word form is then segmented into one or more tokens.

The process of segmenting a textual string into tokens or that of combining them into word forms can be represented in conformance to MAF:

```
<s xml:id="s1" lang="kr">미아가 탔던 차</s>
<maf>
  <token xml:id="s1_tk1">미아</token>
  <!-- mia: 'Mia' -->
  <token xml:id="s1_tk2" join="left">가</token>
  <!-- ga: nominative case marker -->
  <token xml:id="s1_tk3">타</token>
  <!-- tha: 'ride' -->
  <token xml:id="s1_tk4" join="overlap">ㅆ</token>
  <!-- t or ss: past tense makrer -->
  <!-- tk3 and tk4 form one syllable character, 탔: that or thass -->
  <token xml:id="s1_tk5" join="left">던</token>
  <!-- deon: relativizing suffix -->
  <token xml:id="s1_tk6">차</token>
  <!--  cha: 'car' -->
  <wordForm xml:id="s1_wf1" lemma="미아" tag="#pos.properName"
   tokens="s1_tk1" />
  <wordForm xml:id="s1_wf2" lemma="-가" tag="#pos.case_marker"
   tokens="s1_tk2" />
  <wordForm xml:id="s1_wf3" lemma="타다" form="탔던" tag="#pos.verb"
   tokens="s1_tk3  s1_tk4 s1_tk5" />
```

```
  <!-- The lemma 타다 consists of a stem 타 'ride' and a final verbal
   suffix 다. -->
  <wordForm xml:id="s1_wf4" lemma="차" tag="#pos.noun"
   tokens="s1_tk6"/>
</maf>
```

Example 15: Tokens and Word Forms in MAF

4.5.2 Head-final Feature

Korean is an SOV language with predicate forms at the end of the clause. The primary data given above shows a relative clause construction ending in a verb form 탔던 'used to ride' (relativized form) and followed by its nominal head 차 'car'. This syntactic formation can be represented as follows:

```
<s xml:id="s1">
 <graph>
  <terminals>
   <t xml:id="s1_t1" tiger2:corresp="korean.maf.xml#s1_wf1"/>
   <t xml:id="s1_t2" tiger2:corresp="korean.maf.xml#s1_wf2"/>
   <t xml:id="s1_t3" tiger2:corresp="korean.maf.xml#s1_wf3"/>
   <t xml:id="s1_t4" tiger2:corresp="korean.maf.xml#s1_wf4"/>
  </terminals>
  <nonterminals>
   <nt xml:id="s1_nt1" tiger2:type="eojeol" cat="postpositionPhrase">
    <edge xml:id="s1_e1" tiger2:type="noun" tiger2:target="#s1_t1"/>
    <edge xml:id="s1_e2" tiger2:type="case_marker"
     tiger2:target="#s1_t2"/>
   </nt>
   <nt xml:id="s1_nt2" tiger2:type="eojeol" cat="verb">
    <edge xml:id="s1_e3" tiger2:type="wordForm"
     tiger2:target="#s1_t3"/>
   </nt>
   <nt xml:id="s1_nt3" tiger2:type="clause" cat="relativeClause">
    <edge xml:id="s1_e11" tiger2:type="phrase"
     tiger2:target="#s1_nt1"/>
    <edge xml:id="s1_e12" tiger2:type="word" tiger2:target="#s1_nt2"/>
   </nt>
   <nt xml:id="s1_nt4" tiger2:type="eojeol" cat="word">
    <edge xml:id="s1_e4" tiger2:type="noun" tiger2:target="#s1_t4"/>
   </nt>
  </nonterminals>
 </graph>
</s>
```

Example 16: Relative Clause Construction in Korean

5 Generating <tiger2/> data automatically

One of the goals of the <tiger2/> development was not just to provide a new XML format, but also a human and machine readable metamodel and a processable API. For this purpose we used the Eclipse Modeling Framework (EMF). We generated the API in Java, because of its wide use and operating system interoperability. Since the core of the <tiger2/> metamodel is a general graph structure of nodes, edges and labels on both, it is possible, to access the <tiger2/> model with the usual graph processing mechanisms such as traversal etc.

Another part of the <tiger2/> development was its compatibility to the TigerXML model, i.e making sure that existing data in TigerXML can be converted to <tiger2/> without loss of information. To test this, we implemented a mapping from TigerXML to the <tiger2/> API. This allows us to import already existing data in TigerXML into the API and export it to <tiger2/>, and vice versa, though this may lead to information losses in some cases. Testing native TigerXML files and converting them back and forth, we can ensure no information is lost.

Another interesting prospect is converting data from other graph-based formats into <tiger2/>. We therefore used the SaltNPepper framework [15], a universal importer to convert a wide range of formats into each other. Pepper is a plug-in based converter which uses the intermediate metamodel Salt to make a direct conversion between several formats. Using SaltNPepper and the <tiger2/> API we created a module mapping <tiger2/> to the Salt metamodel which we plugged into the framework. This step has allowed us to benefit from all already existing SaltNPepper modules and to convert data e.g. from and into the CoNLL (http://ilk.uvt.nl/conll/#dataformat) format, the PAULA XML format [16], the GrAF format [17] and more.

The CoNLL format, is a field-based format largely used in international parsing evaluations. It is dependency-based, usually restricted to projective dependencies, with several fields such as FORM, LEMMA, CPOSTAG, POSTAG, FEATS, etc. A conversion to <tiger2/> is rather straightforward, with no need for **nt** elements (see Example 17 and 18).

1	il	il	CL	CLS	_	2	suj	_	_
2	mange	manger	V	V	_	0	root	_	_
3	une	un	D	DET	_	4	det	_	_
4	pomme	pomme	N	NC	_	2	obj	_	_
5	.	.	PONCT	PONCT	_	2	ponct	_	_

Example 17: CoNLL output for "he eats an apple"

```
<t form="il" lemma="il" cpostag="CL" postag="CLS" xml:id="1">
 <edge label="suj" target="2"/>
</t>
<t form="mange" lemma="manger" cpostag="V" postag="V"/>
<!-- ….-->
```

Example 18: Fragment of a possible <tiger2/> representation for CoNLL

When using the CoNLL format, some problems may arise from the fact that it does not follow the two level segmentation model of MAF (with tokens and word forms), leading to compound POS such as **P+D** for the agglutinate *des* ('de' + 'les' = *of the*).

In a second experiment, a dataset was converted from the Passage format [18] produced by the French parser FRMG [19][4] into the <tiger2/> format. Though there is no module for SaltNPepper supporting the Passage format at present, a dedicated <tiger2/> generator was constructed for this purpose. The Perl conversion script was easily developed, and, while it remains to test it on complex cases, it demonstrates the potential of <tiger2/> to encode relatively complex syntactic annotation formats.

The Passage format was developed for the French evaluation campaign also called Passage (http://atoll.inria.fr/passage/). Accordingly, the Passage format is currently produced by several French parsers, and is relatively rich in information. It is a successor of the EASy format, extended to better conform the recommendations of ISO TC37SC4, in particular with the MAF format (at the tokenization level with **T** and **W** elements) and with the use of ISO data categories. Passage is both constituency-based, with 6 kinds of chunks (called **G**) and dependency-based, with 14 kinds of relations anchored by either word forms **W** or chunks **G**. The **T** and **W** elements are straightforwardly converted into MAF elements and a correspondence (with attribute @corresp) is established at the level of <tiger2/> elements **t** towards the MAF word forms. The chunks become **nt** elements and the relations become **edge** elements either within **t** or **nt** elements. However, strictly speaking, Passage relations are not oriented (since they entail no explicit notion of governor and governee), but rolebased. Furthermore, the **COORD** relation (for the coordinations) is ternary and has 3 roles, namely **coordonnant** (coordinator), **coord-g** (left coordinated) and **coord-d** (right coordinated). To fit in the <tiger2/> metamodel, it was therefore needed to orient the relations (by choosing a governor and orienting the edge from the governee to its governor) and binarize the COORD relation (using the coordinator as governor). Fortunately, no information is lost in the process. This is achieved using a @**label** attribute on **edge**, which combines the name of the relation with a role name (such as the **SUJ-V_sujet** label for the subject in a SUJ-V relation). Constituency is represented by edges labelled **comp** within **nt** elements. Finally, the additional information carried by Passage **W** elements, such as **form**, **lemma** and **mstag** are moved to the MAF **wordForm** elements, with a conversion of the **mstag** flat feature structure (using tags) into a deep expanded feature structure (based on the FSR standard). Example 21 shows the original Passage fragment and Example 19 and 20 the corresponding representation in MAF and <tiger2/>.

---

[4] FRMG may be tried on line at http://alpage.inria.fr/parserdemo with the possible production of 3 formats (DepXML,Passage, CoNLL). The <tiger2/> format should be soon added, thanks to the script presented in this paper.

```
<maf>
 <token xml:id="E1T1">ce</token>             <!-- this -->
 <token xml:id="E1T2">matin</token>          <!-- morning -->
 <token xml:id="E1T3">,</token>              <!-- , -->
 <token xml:id="E1T4">il</token>             <!-- he →
 <token xml:id="E1T5">a</token>              <!-- has -->
 <token xml:id="E1T6">rapidemment</token>    <!-- quickly -->
 <token xml:id="E1T7">mangé</token>          <!-- eaten -->
 <token xml:id="E1T8">une</token>            <!-- an -->
 <token xml:id="E1T9">pomme</token>          <!-- apple -->
 <token xml:id="E1T10">rouge</token>         <!-- red -->
 <token xml:id="E1T11">et</token>            <!-- and -->
 <token xml:id="E1T12">une</token>           <!-- a -->
 <token xml:id="E1T13">poire</token>         <!-- pear -->
 <token xml:id="E1T14">.</token>             <!-- . -->
 <wordForm form="ce" lemma="ce" tokens="E1T1" xml:id="E1F1">
  <fs>
   <f name="pos"><symbol value="demonstrativeDeterminer"/></f>
   <f name="dem"><symbol value="plus"/></f>
   <f name="numberposs"><symbol value="minus"/></f>
   <f name="gender"><symbol value="masc"/></f>
   <f name="number"><symbol value="sg"/></f>
  </fs>
 </wordForm>
 <!-- …. -->
</maf>
```

Example 19: Fragment of the MAF part (sample_passage.maf.xml) produced from PASSAGE

```
<?xml version="1.0" encoding="UTF-8"?>
<corpus xmlns="http://korpling.german.hu-berlin.de/tiger2/V2.0.5/"
 xmlns:tiger2="http://korpling.german.hu-berlin.de/tiger2/V2.0.5/"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://korpling.german.hu-berlin.de/tiger2/V2.0.5/
 http://korpling.german.hu-berlin.de/tiger2/V2.0.5/Tiger2.xsd">
 <head>
  <meta>
   <name>conversion from Passage format</name>
   <author>passage2tiger2.pl</author>
   <date>2012-08-28 18:25:11</date>
   <format>PASSAGE format with French tagset</format>
  </meta>
  <annotation>
   <external corresp="PassageTAGSET.xml"/>
  </annotation>
 </head>
 <body>
  <s xml:id="s1">
   <graph>
    <terminals>
     <!-- this-->
     <t tiger2:corresp="sample_passage.maf.xml#E1F1"
      xml:id="E1F1"/>
     <!-- morning -->
     <t tiger2:corresp="sample_passage.maf.xml#E1F2" xml:id="E1F2">
```

```
      <edge label="MOD-V_modifieur" tiger2:target="E1F7"
       tiger2:type="prim"/>
     </t>
     <!-- ... -->
     <!--eaten -->
     <t tiger2:corresp="sample_passage.maf.xml#E1F7"
      xml:id="E1F7"/>
     <!-- an-->
     <t tiger2:corresp="sample_passage.maf.xml#E1F8"
      xml:id="E1F8"/>
     <!-- apple-->
     <t tiger2:corresp="sample_passage.maf.xml#E1F9" xml:id="E1F9">
      <edge label="COORD_coord-g" tiger2:target="E1F11"
       tiger2:type="prim"/>
     </t>
     <!-- red -->
     <t tiger2:corresp="sample_passage.maf.xml#E1F10"
      xml:id="E1F10">
      <edge tiger2:target="E1F9" label="MOD-N_modifieur"
       tiger2:type="prim"/>
     </t>
     <!-- and -->
     <t corresp="sample_passage.maf.xml#E1F11" xml:id="E1F11">
      <edge label="COD-V_cod" tiger2:target="E1F7" tiger2:type="prim"/>
     </t>
     <!-- a -->
     <t corresp="sample_passage.maf.xml#E1F12" xml:id="E1F12"/>
     <!-- pear -->
     <t corresp="sample_passage.maf.xml#E1F13" xml:id="E1F13">
      <edge label="COORD_coord-d" tiger2:target="E1F11"
       tiger2:type="prim"/>
     </t>
     <!-- ... -->
    </terminals>
    <nonterminals>
     <nt cat="GN" xml:id="E1G1">
      <edge label="comp" tiger2:target="E1F1" tiger2:type="prim"/>
      <edge label="comp" tiger2:target="E1F2" tiger2:type="prim"/>
     </nt>
     <!-- ... -->
    </nonterminals>
   </graph>
  </s>
 </body>
</corpus>
```

Example 20: Fragment of the <tiger2/> part (sample_passage.tiger2.xml) produced from PASSAGE and relying on the MAF part and an external PASSAGE tagset (PassageTAGSET.xml) , which specifies the possible features (pos, cat, lemma, form, …) and values (in particular for the edge labels and for the partof-speech

```
<Document dtdVersion="2.0" xmlns:xlink="http://www.w3.org/1999/xlink">
 <Sentence id="E1" trust="100">
  <T id="E1T1">ce</T>
  <T id="E1T2">matin</T>
```

```
  <G id="E1G1" type="GN">
   <W id="E1F1" pos="demonstrativeDeterminer" lemma="ce" form="ce"
    tokens="E1T1" mstag="wh.minus dem.plus numberposs.minus gender.masc
    number.sg"/>
   <W id="E1F2" pos="commonNoun" lemma="matin" form="matin"
    tokens="E1T2" mstag="person.3 wh.minus time.arto hum.minus
    gender.masc number.sg"/>
  </G>
  <!-- ... -->
  <R type="MOD-V" id="E1R5">
   <modifieur ref="E1F2"/>
   <verbe ref="E1F7"/>
  </R>
  <R type="COORD" id="E1R8">
   <coordonnant ref="E1F11"/>
   <coord-g ref="E1F9"/>
   <coord-d ref="E1F13"/>
  </R>
 </Sentence>
</Document>
```

Example 21: Fragment of the original PASSAGE file used to produce the MAF and <tiger2/> parts

6 Next steps

Even if the <tiger2/> proposal, building upon the already quite mature TigerXML format, does already present all the technical features to cover the characteristics of the SynAF metamodel, making it an ISO standard, usable for the largest possible scientific community will require some additional efforts. In particular, beyond the provision of simple examples, we will need to put <tiger2/> to the test by confronting it with large-scale treebanks as they are available in the various languages tackled in this paper. In particular, this will bring more ideas as to the data categories that should be added to ISOCat in order to get a fully-fledged environment for the description of annotation schemes for syntactic data.

References

[1] ISO 24615:2010. Language resource management – Syntactic annotation framework (SynAF).

[2] König, E. & Lezius, W. (2000). The TIGER language – A Description Language for Syntax Graphs. In: *Proceedings of COLING 2000*, 1056–1060.

[3] Leech G. N., Barnett, R. & Kahrel, P. (1995). *EAGLES Final Report and Guidelines for the Syntactic Annotation of Corpora*. EAGLES Document EAGTCWG-SASG. Pisa, Italy.

[4] Ide, N. & Romary, L. (2003). Encoding Syntactic Annotation. In: A. Abeillé (Ed.) *Treebanks*, Springer [available online at http://hal.archives-ouvertes.fr/hal-00079163].

[5] ISO/FDIS 24611. Language resource management – Morpho-syntactic annotation framework (MAF).

[6] Alarcos-Llorach, E. (2004[1999]). *Gramática de la lengua española*. Madrid: Espasa Libros, S.L.U.

[7] Real Academia Española (2009). *Nueva gramática de la lengua española*. Madrid: Espasa Libros, S.L.U.

[8] Pineda, L. & Meza, I. (2005). The Spanish pronominal clitic system. In: *SEPLN (Revista de la Sociedad Española para el Procesamiento del Lenguaje Natural)* 34 [available online at http://www.sepln.org/revistaSEPLN/revista/34/06.pdf (visited on 28 August 2012)].

[9] Ryding, K. C. (2005). *A Reference Grammar of Modern Standard Arabic*. Cambridge: Cambridge University Press.

[10] Borer, H. (2008). Compounds: The View from Hebrew. In: R. Lieber & P. Štekauer (eds.), *The Oxford Handbook of Compounds.* Oxford: Oxford University Press, 491–511.

[11] Nurse, D. & Philippson, G. (2003). *The Bantu Languages*. London: Routledge.

[12] Bresnan, J. (2000). *Lexical-Functional Syntax*. Blackwell Publishing: MA.

[13] Chang, Suk-Jin (1996). *Korean*. Amsterdam: John Benjamins Publishing Co.

[14] Sohn, H.-M. (1999). *The Korean Language*. Cambridge: Cambridge University Press.

[15] Zipser, F., & Romary, L. (2010). A Model Oriented Approach to the Mapping of Annotation Formats Using Standards. In *Proceedings of the Workshop on Language Resource and Language Technology Standards, LREC 2010*. Malta, 7-18.

[16] Dipper, S. (2005). XML-based Stand-off Representation and Exploitation of Multi- Level Linguistic Annotation. In *Proceedings of Berliner XML Tage 2005 (BXML 2005)*. Berlin, Germany, 39-50.

[17] Ide, N., & Suderman, K. (2007). GrAF: A Graph-based Format for Linguistic Annotations. In *Proceedings of the Linguistic Annotation Workshop 2007*, Prague, 1-8.

[18] Paroubek, P., Villemonte de la Clergerie, E., Loiseau, S., Vilnat, A., Francopoulo. G (2009). The PASSAGE syntactic representation. In : *7th International Workshop on Treebanks and Linguistic Theories (TLT7)*. Groningen.

[19] Villemonte de la Clergerie, É. (2010). Convertir des dérivations TAG en dépendances. In *ATALA, Ed., 17e Conférence sur le Traitement Automatique des Langues Naturelles - TALN 2010*.